Evaluating Robustness of LLMs to Numerical Variations in Mathematical Reasoning

Yuli Yang[1] Hiroaki Yamada[1] Takenobu Tokunaga[1]
[1]Institute of Science Tokyo
{yang.y.aw@m, yamada@c, take@c}.titech.ac.jp

Abstract

Evaluating an LLM's robustness against numerical perturbation is a good way to know if the LLM actually performs reasoning or just replicates patterns learned. We propose a novel method to augment math word problems (MWPs), producing numerical variations at a large scale utilizing templates. We also propose an automated error classification framework for scalable error analysis, distinguishing calculation errors from reasoning errors. Our experiments using the methods show LLMs are weak against numerical variations, suggesting they are not fully capable of generating valid reasoning steps, often failing in arithmetic operations.

1 Introduction

Recent LLMs [1, 2, 3, 4] have reported high accuracy rates on mathematical reasoning benchmarks, including GSM8K and MATH [5, 6]. However, a natural concern is that the models just follow surface patterns observed in their pretraining data rather than performing mathematical reasoning [7, 8, 9, 10, 11].

Perturbing superficial elements like names of individuals or specific numbers does not change how the problem should be solved. If an LLM can perform reasoning in solving a math question, it should give correct answers with similar reasoning steps for both the question and its perturbed one. Recent studies [12, 13, 14, 15] evaluated models' robustness against the perturbations based on this hypothesis.

These studies have the following limitations: a) the size of the introduced variations was limited, b) they did not discuss ranges of numerical values such as digit sizes, and c) they did not distinguish reasoning errors and computational errors and could not explain the source of errors.

To address the limitations, we propose a scalable method to augment a math word problem (MWP) dataset by changing numerical values based on template questions. To analyze the impact of digit sizes on models' mathematical reasoning, we controlled the range of the replaced values and generated two distinct subsets, one with questions containing a small number of digits (1-99) and one with questions containing a large number of digits (1-9,999). Using our method, we constructed a new dataset, GSM-ALT, generating 2,000 variants for each original question from GSM8K. Moreover, we propose a novel framework for automated error analysis to identify whether a source of incorrect prediction stems from errors in logical reasoning or numerical calculation.

2 Related Work

Despite strong performance on math benchmarks, researchers are questioning whether current benchmarks can adequately evaluate reasoning abilities and language models demonstrate them.

Levy [7] expanded questions by adding non-essential contents, showing that models' performance decreases when the number of tokens in a problem increases. PlanBench [8, 9] is a benchmark to evaluate planning and reasoning capabilities. Their findings suggest that even state-of-the-art models still struggle with this. Srivastava [12] functionalized the math questions to create a dynamic dataset, providing a robust evaluation metric against potential data leakage to models' pretraining. Jiang [10] demonstrated that the models' high accuracy depends on a specific token bias, and the models' reasoning capability depends on recognizing certain superficial patterns. Berglund [16] and Guo [11] gave the answer to the questions and reversed to infer one of the variables to construct reversal versions of the original questions. They showed that the current models performed poorly on the reversal ones.

3 Method

3.1 Question Template Development

To develop a new dataset to assess the robustness of numerical variations, we manually generate new variants based on templates (Figure 1) composed from the original questions of an existing dataset.

A question from an existing dataset (e.g., GSM8K) has tuples of (question Q, solution S). Q is a natural language text describing a question to be solved. S contains the process P and the final answer A. P shows a gold process for solving the question Q step by step, including equations. A stores a numerical value as a gold outcome from the P. Given (Q, S), we first replace all the numerical values in the Q with variables to get Q_{abs}, which is the abstracted Q. We apply the same operation to S and get S_{abs}. We keep variables consistent between Q_{abs} and S_{abs}. S_{abs} contains P_{abs} and A_{abs}, representing the abstracted P and A. The Q_{abs} and S_{abs} constitute a question template T.

3.2 Variant Set Generation

Given a template T of an original question, we generate variants by replacing the variables in T with random values. t_i denotes a variant generated from T, consisting of question Q_i, solution S_i. S_i contains the process P_i and final answer A_i. To ensure the variants are valid, the replaced values need to satisfy some constraints (Figure 1). For example, an answer should be positive and whole when it represents the number of objects. Intermediate values appearing in the process P_i also need to satisfy the constraints as well. We manually define constraints for each template. We only accept a variant if it satisfies the constraints.

If models do only superficial pattern-based inference and do not conduct reasoning, they perform poorly in solving questions containing numbers that are rare in their training, such as large digit numbers. To examine this hypothesis, for each question template, we controlled the replaced values within two different ranges and subsequently resulted in two different variant sets: 1-99 (namely, the Easy variant set) and 1-9,999 (namely, the Hard variant set).

4 Experimental Settings

We use GSM8K as the base dataset for our experiment. GSM8K consists of MWPs for primary and secondary school students and involves only the four basic arithmetic operations. We randomly sampled 92 questions from the GSM8K training set, from which we manually created 92 question templates[1)]. Given the templates, we generated 1,000 hard variants and 1,000 easy variants for each template. As a result, our new dataset GSM-ALT consists of the Hard and Easy variant set, each containing 92,000 variants.

We use accuracy as a primary evaluation metric. For the original instances from the base dataset (original GSM8K), we use a standard accuracy. For generated variants from our dataset, we first calculate the accuracy for each template variant set containing 1,000 variants, and then we average them over all 92 templates.

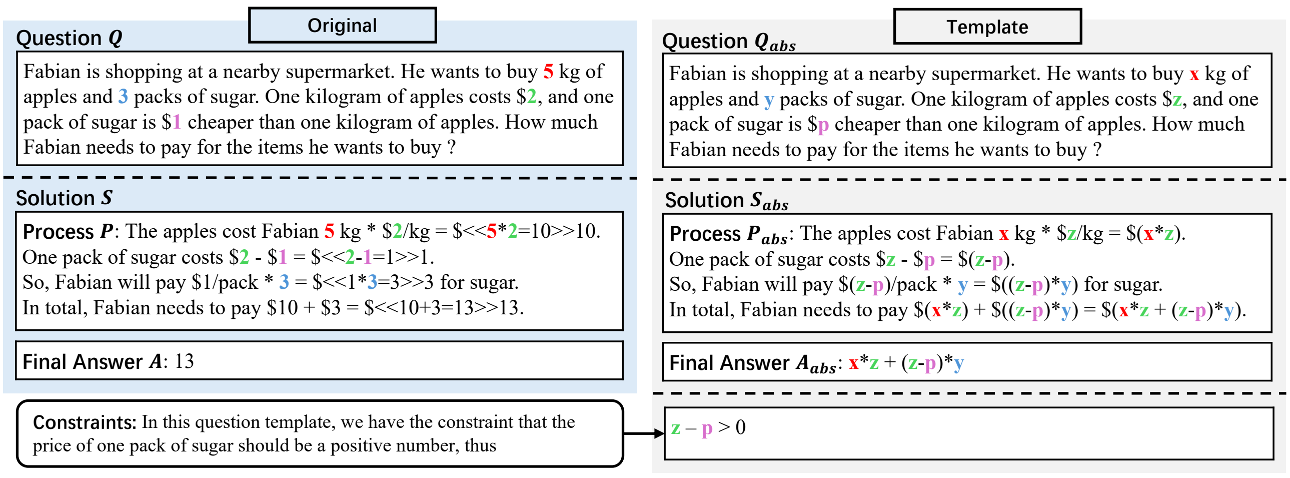
The target models to be evaluated include generic models (Llama-3-8b-Instruct, Llama-3.1-8b-Instruct, Llama-3.1-70b-Instruct, Mistral-7b-Instruct-v0.3) and math models that were fine-tuned on mathematical contents (Deepseekmath-7b-rl, Wizardmath-7b-v1.1)

Regarding the generation settings, we used greedy search to maximize the reproducibility and stability of results. To minimize the influence of few-shot examples while ensuring that the model can perform mathematical reasoning, we adopted the zero-shot Chain-of-Thought (CoT) prompting for solution generation and extracted the final answer in the same way as Kojima [17] for generic models. As for math models, we adopted the specifically designed prompts, which are recommended on their Web pages. The prompts used in the experiment can be found in Appendix C.

5 Results

Table 1 shows the results of each model's accuracy evaluated on the original GSM8K and our GSM-ALT. The lowest scores are highlighted in boldface. GSM-ALT results show scores from the Easy variant set and the Hard variant set. All models showed a significant performance drop in GSM-ALT from the base GSM8K. The drop was observed in both the Easy and the Hard variant sets. Even the two math-specialized models, especially Wizardmath-

1) We initially created 250 templates but the number of possible variants is limited in some of the templates, so we removed those templates to ensure there is no duplicated variant


Original

Question Q

Fabian is shopping at a nearby supermarket. He wants to buy 5 kg of apples and 3 packs of sugar. One kilogram of apples costs \$2, and one pack of sugar is \$1 cheaper than one kilogram of apples. How much Fabian needs to pay for the items he wants to buy ?

Solution S

Process P: The apples cost Fabian 5 kg * \$2/kg = \$<<5*2=10>>10.
One pack of sugar costs \$2 - \$1 = \$<<2-1=1>>1.
So, Fabian will pay \$1/pack * 3 = \$<<1*3=3>>3 for sugar.
In total, Fabian needs to pay \$10 + \$3 = \$<<10+3=13>>13.

Final Answer A: 13

Constraints: In this question template, we have the constraint that the price of one pack of sugar should be a positive number, thus

Template

Question Q_{abs}

Fabian is shopping at a nearby supermarket. He wants to buy x kg of apples and y packs of sugar. One kilogram of apples costs \$z, and one pack of sugar is \$p cheaper than one kilogram of apples. How much Fabian needs to pay for the items he wants to buy ?

Solution S_{abs}

Process P_{abs}: The apples cost Fabian x kg * \$z/kg = \$(x*z).
One pack of sugar costs \$z - \$p = \$(z-p).
So, Fabian will pay \$(z-p)/pack * y = \$((z-p)*y) for sugar.
In total, Fabian needs to pay \$(x*z) + \$((z-p)*y) = \$(x*z + (z-p)*y).

Final Answer A_{abs}: x*z + (z-p)*y

$z - p > 0$

Figure 1 Example of Question Template Development

Table 1 Accuracy scores

Models	GSM8K	GSM-ALT	
	Base	Easy	Hard
Llama-3-8b-Instruct	0.840	0.507	**0.156**
Llama-3.1-8b-Instruct	0.880	0.604	**0.193**
Llama-3.1-70b-Instruct	0.978	0.819	**0.355**
Mistral-7b-Instruct-v0.3	0.587	0.238	**0.104**
Deepseek-math-7b-rl	0.957	0.706	**0.307**
Wizardmath-7b-v1.1	0.891	0.489	**0.223**

7b-v1.1, showed lower scores by more than 0.4 on the Easy and more than 0.6 on the Hard.

This result shows that numerical variations always degrade performance in both the Hard and the Easy variant sets. The fact that the Easy variant set degrades the performance indicates that the models are weak even against the numbers whose range is similar to the base GSM8K. Moreover, we found clearer score drops from the GSM8K scores in the Hard variant set than in the Easy variant set, suggesting the computational difficulty affects models' reasoning.

6 Error Analysis on Solutions

To identify the source of errors, we classify errors into two types: calculation errors and reasoning errors. If an incorrect solution only contains failures in calculations, we call it a calculation error. If an incorrect solution contains incorrect reasoning steps, we label it a reasoning error regardless of its incorrect calculations.

As GSM-ALT will be larger than its original dataset, manually checking each generated solution is not practical, and thus, we propose a novel framework that automatically classifies errors into calculation or reasoning errors.

6.1 Error Analysis Framework

To classify errors, we first transform a predicted solution $\hat{S}_i$ into its abstracted form $\hat{S}^i_{abs}$, which contains the abstracted $\hat{P}^i_{abs}$ and $\hat{A}^i_{abs}$. If $\hat{S}_i$ is incorrect because of a reasoning error, its transformed $\hat{P}^i_{abs}$ should contain a reasoning error resulting in incorrect $\hat{A}^i_{abs}$. If $\hat{S}_i$ contains a calculation error, but its reasoning steps are correct, $\hat{P}^i_{abs}$ and $\hat{A}^i_{abs}$ should be correct. Thus, checking if $\hat{A}^i_{abs}$ is correct should give a proxy to determine the sources of errors.

In our framework, an LLM transforms a $\hat{S}_i$ into the $\hat{S}^i_{abs}$, as shown in Figure 2. Then, we can automatically check if $\hat{A}^i_{abs}$ is correct by comparing it with its gold answer A_{abs} from our templates. An input to the LLM is a model's predicted solution $\hat{S}_i$, its question Q_i, and its abstracted question Q_{abs} available from our templates. We auxiliaryly input the Q_{abs} guiding the LLM to use variables consistently, inspired by Gaur [18]. An output from the LLM is an abstracted solution $\hat{S}^i_{abs}$. We show our prompt for this framework in Appendix D. We employ Qwen2-math-72b-instruct [19] for this transformation. We manually checked the outputs and confirmed the LLM could obtain the abstracted solutions at 90% success rate on average in our preliminary experiment.

6.2 Results of Error Analysis

Table 2 shows the results of error classification by our framework. Values in the table indicate the proportion of

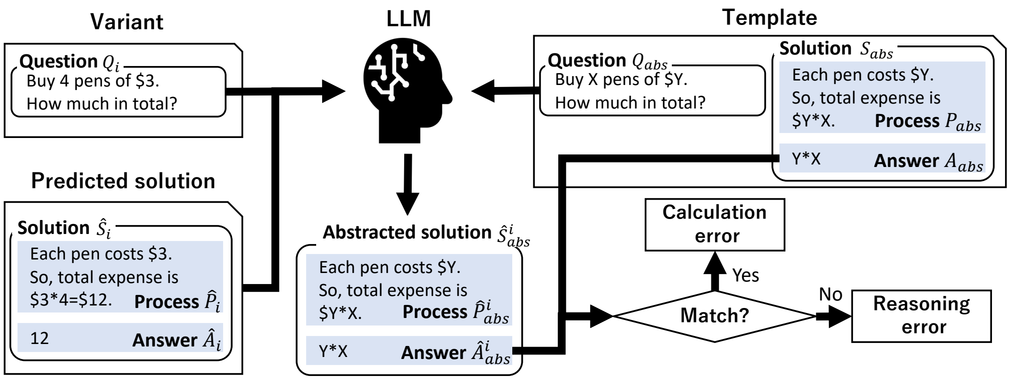


Figure 2 Error classification framework

Table 2 Error rate per error type and variant set

	Base set		Easy variant set		Hard variant set	
	calculation err.	reasoning err.	calculation err.	reasoning err.	calculation err.	reasoning err.
Llama-3-8b-Instruct	.033 (20.0%)	.130 (80.0%)	.279 (56.6%)	.214 (43.4%)	.573 (67.9%)	.271 (32.1%)
Llama-3.1-8b-Instruct	.033 (27.3%)	.087 (72.7%)	.252 (63.6%)	.144 (36.4%)	.601 (74.5%)	.206 (25.5%)
Llama-3.1-70b-Instruct	.000 (00.0%)	.022 (100.0%)	.125 (69.1%)	.056 (30.9%)	.516 (80.0%)	.129 (20.0%)
Mistral-7b-Instruct-v0.3	.098 (23.7%)	.315 (76.3%)	.385 (50.5%)	.377 (49.5%)	.477 (53.2%)	.419 (46.8%)
Deepseek-math-7b-rl	.011 (25.0%)	.033 (75.0%)	.217 (73.8%)	.077 (26.2%)	.529 (76.4%)	.163 (23.6%)
Wizardmath-7b-v1.1	.043 (40.0%)	.065 (60.0%)	.383 (75.0%)	.128 (25.0%)	.586 (75.4%)	.191 (24.6%)
Macro avg.	.036 (24.8%)	.109 (75.2%)	.274 (62.3%)	.166 (37.7%)	.547 (70.4%)	.230 (29.6%)

solutions classified as calculation errors or reasoning errors out of all solutions predicted by the models. Values in parentheses indicate the proportion of solutions classified as calculation errors or reasoning errors out of incorrect solutions.

In the Base set, the majority of incorrect solutions were due to reasoning errors, while they changed to calculation errors in the Easy and Hard variant sets. This trend was especially evident in the Hard variant set, and more than 70% were because of calculation errors. This result suggests that the limited capability of arithmetic calculation is indeed a major issue of LLMs in solving mathematical problems rather than the reasoning capability of generating a valid process of solving steps especially when the numerical values in the questions are large.

Looking at the reasoning errors, all the models got more errors in both the Easy and Hard variant sets than the base set. The same as calculation errors, the trend was evident in the Hard variant set. This result suggests that variants also introduce harmful changes in reasoning steps in addition to complex calculations, which result in incorrect solutions. Moreover, variants with larger digit sizes are more likely to introduce errors in reasoning steps.

7 Conclusion

We proposed a novel method to augment MWP datasets, which produces a dataset for evaluating LLMs' robustness against numerical variations at a reliable scale. Using our templates, anyone can easily generate thousands of variants from one original question in the GSM8K, which was not possible with any preceding proposals. We also proposed an automated error classification framework for scalable error analysis, distinguishing calculation errors from reasoning errors.

Using the methods, we empirically showed that the six LLMs we tested were weak against numerical variations, especially when the numerical values were large. This finding is consistent with previous studies [12, 13, 14, 15], but we confirm it with more variants. Our error analysis uniquely identified that calculation errors contributed to a substantial proportion of incorrect solutions, suggesting LLMs' incapability of arithmetic operations is the main source of limited capabilities in math word problems. Moreover, we found that LLMs still fail in their reasoning steps, especially when they encounter variants with larger numerical values. Given our findings, it is still hard to say that current LLMs are robust against numerical variations.

Acknowledgement

This work was partly supported by JST, PRESTO Grant Number JPMJPR236B, Japan.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. **arXiv preprint arXiv:2303.08774**, 2023.

[2] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. **arXiv preprint arXiv:2407.21783**, 2024.

[3] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. **arXiv preprint arXiv:2312.11805**, 2023.

[4] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. **arXiv preprint arXiv:2408.00118**, 2024.

[5] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021.

[6] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. **arXiv preprint arXiv:2103.03874**, 2021.

[7] Mosh Levy, Alon Jacoby, and Yoav Goldberg. Same task, more tokens: the impact of input length on the reasoning performance of large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, **Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pp. 15339–15353, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[8] Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. **Advances in Neural Information Processing Systems**, Vol. 36, , 2024.

[9] Karthik Valmeekam, Kaya Stechly, and Subbarao Kambhampati. Llms still can't plan; can lrms? a preliminary evaluation of openai's o1 on planbench, 2024.

[10] Bowen Jiang, Yangxinyu Xie, Zhuoqun Hao, Xiaomeng Wang, Tanwi Mallick, Weijie J Su, Camillo Jose Taylor, and Dan Roth. A peek into token bias: Large language models are not yet genuine reasoners. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, **Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing**, pp. 4722–4756, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[11] Pei Guo, WangJie You, Juntao Li, Yan Bowen, and Min Zhang. Exploring reversal mathematical reasoning ability for large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, **Findings of the Association for Computational Linguistics: ACL 2024**, pp. 13671–13685, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[12] Saurabh Srivastava, Annarose M B, Anto P V, Shashank Menon, Ajay Sukumar, Adwaith Samod T, Alan Philipose, Stevin Prince, and Sooraj Thomas. Functional benchmarks for robust evaluation of reasoning performance, and the reasoning gap, 2024.

[13] Kun Qian, Shunji Wan, Claudia Tang, Youzhi Wang, Xuanming Zhang, Maximillian Chen, and Zhou Yu. VarBench: Robust language model benchmarking through dynamic variable perturbation. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, **Findings of the Association for Computational Linguistics: EMNLP 2024**, pp. 16131–16161, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[14] Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. GSM-plus: A comprehensive benchmark for evaluating the robustness of LLMs as mathematical problem solvers. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, **Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pp. 2961–2984, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[15] Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. **arXiv preprint arXiv:2410.05229**, 2024.

[16] Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. The reversal curse: Llms trained on "a is b" fail to learn "b is a", 2024.

[17] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. **Advances in neural information processing systems**, Vol. 35, pp. 22199–22213, 2022.

[18] Vedant Gaur and Nikunj Saunshi. Reasoning in large language models through symbolic math word problems, 2023.

[19] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. **arXiv preprint arXiv:2407.10671**.

A Necessity of Manual Operations in Creating Question Templates

Although we have considered using regular expressions and rule-based approaches to automate template creation, they have the following problems: a) Not all numerical values in the original instance are "symbolizable." Some numbers in the instance are specific, and altering them would make the instance ill-defined. b) As shown in Figure 1, when generating the template, it is necessary to keep the usage of variable consistent between Q_{abs} and S_{abs}. It is hard to catch the relationship with rule-based replacement and requires human insight. Therefore, we created the question templates manually.

B Large Language Models

We list all of the LLMs used in our experiments.

Generic LLMs

- Llama-3-8b-Instruct (`https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct`)
- Llama-3.1-8b-Instruct (`https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct`)
- Llama-3.1-70b-Instruct (`https://huggingface.co/meta-llama/Llama-3.1-70B-Instruct`)
- Mistral-7b-Instruct-v0.3 (`https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3`)

LLMs for mathematical domain

- Deepseekmath-7b-rl (`https://huggingface.co/deepseek-ai/deepseek-math-7b-rl`)
- Wizardmath-7b-v1.1 (`https://huggingface.co/WizardLMTeam/WizardMath-7B-V1.1`)

C Prompts Design for Main Experiments

For the generic LLMs, we developed prompts for solution generation (Figure 3) and answer extraction (Figre 4) based on the prompts used in [17].

Generation Prompt – generic models

SYSTEM: You are an assistant that solves math word problems.

USER: {question} + Let's think step by step.

Figure 3 The prompt for generic models (generating solutions)

Answer Extraction Prompt – generic models

SYSTEM: You are an assistant that solves math word problems.

USER: {question} + Let's think step by step.
ASSISTANT: {model's completion}

USER: Therefore, what is the final answer? Only write the final answer without any texts.

Figure 4 The prompt for generic models (extracting final answer)

For Deepseekmath-7b-rl and Wizardmath-7b-v1.1, we employed prompts based on templates suggested on their web pages. Figure 5 and 6 show them. In extracting answers from solutions generated by the two math models, we could simply use regular expressions since they always generate solutions in a fixed format.

Generation Prompt – Deepseekmath-7b-rl

USER: {question}
Please reason step by step and put your final answer within \boxed{}.

Figure 5 The prompt for Deepseekmath-7b-rl (genearting solutions)

Generation Prompt – Wizardmath-7b-v1.1

USER: Below is an instruction that describes a task. Write a response that appropriately completes the request.
Instruction:
{question}
Response:
Let's think step by step.

Figure 6 The prompt for Wizardmath-7b-v1.1 (generating solutions)

D Prompt Design for Error Analysis Framework

Figure 7 presents the prompt used to transform a predicted solution into the abstracted form.

Transformation Prompt

SYSTEM: Given the numeric version of a math question and its solution as references, you are a helpful assistant designed to copy the numeric solution to get a solution to the symbolic version of that question.
Instructions:
- Symbolic solution should strictly copy the numeric solution no matter whether it is correct or not.
- After completion of the solution, output the final answer with "###". The final answer should be a sole mathematical expression represented by variables appear in the symbolic question.
- Mathematical expression in the symbolic solution should not be represented in the format of LaTeX.

{few-shot examples}

USER: {target solution}

Figure 7 The prompt for solution transformation